6. Judy

Program Name: judy.java Input File: judy.dat

Judy has just learned that an array can be used to store a binary tree data structure, where the root node is stored in position 0, and the left and right children are stored in positions 1 and 2. The left and right children of the root's left child are in positions 3 and 4, with positions 5 and 6 containing the two children of the root's right child. This goes on and on, with the left child always in a node based on its parent's position * 2 + 1, and the right child in parent's position * 2 + 2.

Consider the tree below and the array diagram and study how this works. The root C is in position zero, and its left child J is in position (0 * 2 + 1), or 1, and the right child D is in position (0 * 2 + 2), or 2. The children of M, which is in position 6, are in positions 13 (6 * 2 + 1) and 14 (6 * 2 + 2).

```
    C
   / \
  J   D
 / \   \
K   L   M
       / \
      N   A
```

The resulting array representation for this is shown here:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
C J D K L – M – – –  –  –  –  N  A
```

To challenge herself, Judy decided to list on a line of data each parent/left child/right child triple as a cluster of three characters, with the parent of the triple listed first, followed by the left child, and then the right child. The root triple in the diagram would be **`CJD`**. The rest of the triples would include **`JKL, D-M,`** and **`MNA.`** The dash means there is no child in that position.

The challenge would be to list the triples in random order, put them all on one line in a data file, with a single space between each triple, and then write a program to sort it all out, putting the triples into their proper positions in the array representation of the resulting binary tree.

To test this challenge, she always included the node **`A`** once somewhere in the tree, and then tried to figure in what position it ended up in the array. For example, if the line of triples was in this order **`- MNA CJD JKL D-M`** – she would first need to find the root triple, but she discovered that was easy since the parent letter of the root triple doesn't appear in the 2nd or 3rd position of any other triple. The C of the triple CJD does not appear in any of the other triples, and so it is clearly the root node. The rest of the tree is fairly easy to figure out, working down from the root.

Input: Several data sets, each on one line consisting of a series of binary tree triples, in no certain order.

Output: The final position of the node **`A`** in the array representation of the resulting binary tree.

Sample input:
```
MNA CJD JKL D-M
SAL ATR
HB- BA- ADE
```
Sample output:
```
14
1
3
```